

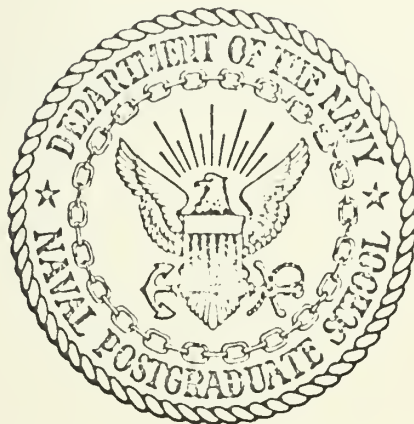
AN INTERFACE FOR THE PDP-8 COMPUTER SYSTEM  
COMPRISING ASSEMBLY, COMPILATION, SIMULATION,  
AND PDP-8 EXECUTION OF  
RESULTING OBJECT MODULES

John Winthrop Barnes



# NAVAL POSTGRADUATE SCHOOL

## Monterey, California



# THESIS

An INTERFACE for the PDP-8 Computer System  
Comprising Assembly, Compilation, Simulation,  
and  
PDP-8 Execution of Resulting Object Modules

*Thesis*  
*B2297*

by

John Winthrop Barnes, Jr.

Thesis Advisor:

Alan B. Roberts

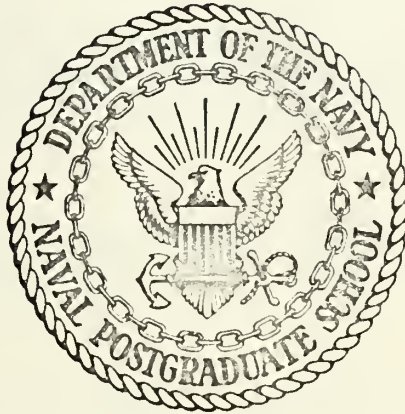
June 1972

*Approved for public release; distribution unlimited.*



# NAVAL POSTGRADUATE SCHOOL

## Monterey, California



# THESIS

An INTERFACE for the PDP-8 Computer System  
Comprising Assembly, Compilation, Simulation,  
and  
PDP-8 Execution of Resulting Object Modules

by

John Winthrop Barnes, Jr.

Thesis Advisor:

Alan B. Roberts

June 1972

*Approved for public release; distribution unlimited.*



An INTERFACE for the PDP-8 Computer System  
Comprising Assembly, Compilation, Simulation,  
and  
PDP-8 Execution of Resulting Object Modules

by

John Winthrop Barnes, Jr.  
Lieutenant, United States Navy  
B. S., United States Naval Academy, 1966

Submitted in partial fulfillment of the  
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL  
June 1972





## ABSTRACT

The design and implementation of the INTERFACE for the PDP-8 Computer System is described. The INTERFACE executes on an IBM System 360 and allows PAL III assembly, FORTRAN/8 compilation, and simulated execution of the resulting object code. If desired, object code is generated and processed for later execution on a PDP-8 computer.



## TABLE OF CONTENTS

|      |  |    |
|------|--|----|
| I.   | INTRODUCTION -----                                   | 5  |
| A.   | STATEMENT OF THE PROBLEM -----                       | 5  |
| 1.   | Need for Simulator -----                             | 6  |
| 2.   | Search for Previous Work -----                       | 6  |
| B.   | REDEFINITION OF THE PROBLEM -----                    | 7  |
| II.  | A DESCRIPTION OF THE INTERFACE -----                 | 8  |
| A.   | INTERFACING THE COMPILER AND SIMUL8S -----           | 8  |
| B.   | INTERFACING THE OBJECT MODULE AND SIMUL8S -----      | 10 |
| C.   | INTERFACING THE OBJECT MODULE AND PDP-8 -----        | 12 |
| III. | INTERFACE TEST RESULTS -----                         | 14 |
| A.   | SIMULATED EXECUTION OF FORTRAN/8 OBJECT MODULE ----- | 14 |
| B.   | PDP-8/S OBJECT MODULE EXECUTION-----                 | 15 |
| 1.   | Compiled Module -----                                | 15 |
| 2.   | Assembled Modules -----                              | 15 |
| C.   | PDP-8/E OBJECT MODULE EXECUTION -----                | 15 |
| 1.   | Compiled Modules-----                                | 16 |
| 2.   | Assembled Modules -----                              | 16 |
| IV.  | PROBLEM AREAS -----                                  | 17 |
| A.   | RESOLVED -----                                       | 17 |
| 1.   | Assembler -----                                      | 17 |
| 2.   | Simulator -----                                      | 18 |
| B.   | UNRESOLVED -----                                     | 19 |
| 1.   | Compiler -----                                       | 19 |



|   |    |
|---|----|
| V. CHANGES MADE TO THE SIMUL8S PACKAGE -----  | 21 |
| A. GENERAL CHANGES -----                      | 21 |
| 1. PDP-8/E Simulation -----                   | 21 |
| 2. Teletype Input -----                       | 21 |
| 3. EAE Simulation -----                       | 21 |
| B. SPECIFIC SUBROUTINE CHANGES -----          | 22 |
| VI. RECOMMENDATIONS -----                     | 23 |
| A. ZEROING OF PDP-8 MEMORY -----              | 23 |
| B. COMPILER CHANGES -----                     | 23 |
| C. PI-1250-1 INTERFACE -----                  | 24 |
| D. HAND PLACEMENT OF CHECKSUM -----           | 25 |
| VII. CONCLUSIONS -----                        | 26 |
| APPENDIX A: INTERFACE USER MANUAL -----       | 27 |
| APPENDIX B: JOB CONTROL LANGUAGE -----        | 30 |
| APPENDIX C: CDC-160 PROGRAM LISTING -----     | 32 |
| APPENDIX D: CDC-160 MEMORY MAP -----          | 38 |
| APPENDIX E: SIMUL8S USER MANUAL CHANGES ----- | 39 |
| LIST OF REFERENCES -----                      | 40 |
| INITIAL DISTRIBUTION LIST -----               | 41 |
| FORM DD 1473 -----                            | 42 |



## I. INTRODUCTION

At the Naval Postgraduate School there are presently two computers of the PDP-8 series.<sup>1</sup> The Oceanographic Department is assigned a system consisting of a PDP-8/S computer, ASR 33 teletype, and a PI-1250-1 Data Handling System. The Operations Research/Administrative Sciences Department is assigned a system consisting of a PDP-8/E computer, ASR 33 teletype, and a DECtape Transport Unit.

### A. STATEMENT OF THE PROBLEM

There is at present a widespread use of minicomputers throughout industry, business, and educational institutions. The PDP-8 series of computers is representative of this group. In general these minicomputers are used for production program execution and in this regard they perform very satisfactorily. However, in program development and debugging they are sorely lacking. This is due primarily to the small memory, typically four to eight K, associated with these machines. The size restriction usually results in only one program being able to reside in core at any given time. This coupled with the slow input/output speed of the teletype results in the user spending an excessive amount of time readying his program for execution.

Another severe limitation due once again to the small memory size is the lack of an operating system resulting in the fact that only one user has access to the system at a time.

---

<sup>1</sup>The PDP-8 series are referred to as PROGRAMMED DATA PROCESSORS and are manufactured by Digital Equipment Corporation, Maynard, Massachusetts.





## 1. Need for Simulator

FORTRAN/8 was written with the preceding discussion in mind.<sup>2</sup>

Both the use of a high-level language in the generation of object code and a large size computer such as the IBM System 360 helped to alleviate the program development problem.

In regard to the program debugging problem, the original concept of this thesis was to write a simulator of the PDP-8 Computer System that would execute the object code generated by FORTRAN/8. This simulator would run on the IBM System 360 thereby enhancing the advantages gained by using FORTRAN/8 because complete program development, including testing and debugging, could be accomplished apart from the PDP-8.

## 2. Search for Previous Work

In order to preclude duplication of effort a search was made to determine if a simulator of this type had been written previously. The search resulted in the Naval Postgraduate School's purchase of the SIMUL8S software package.<sup>3</sup> SIMUL8S is a PAL III assembler for and a simulator of the PDP-8 series of computers.<sup>4</sup> It is written in FORTRAN IV and runs on the IBM System 360.

---

<sup>2</sup>FORTRAN/8 is a FORTRAN compiler written in XPL which runs on the IBM System 360. Its output, a 4K object module, is acceptable for execution on a PDP-8 computer.

<sup>3</sup>SIMUL8S is referred to as Simulation 8 Series and was programmed by Decision Science, Inc., San Diego, California. References 1 and 2 are the user manuals provided by DSI.

<sup>4</sup>PAL III is referred to as Program Assembly Language third version and is the PDP-8 assembler which translates symbolic programs written in PAL III language into binary programs.



## B. REDEFINITION OF THE PROBLEM

Now that a simulator had been found, the question arose as to whether enough work remained to consider a thesis in this area. It was decided that a complete interface between FORTRAN/8, SIMUL8S, and the PDP-8 machines presently existing at the Naval Postgraduate School was a necessity for significant use of the system by the PDP-8 user groups. It was not considered reasonable to expect the users themselves to produce such an interface since it would require manipulating of two separate software packages and working with three different machines. In addition the users comprise two distinct and separate groups each with their own requirements. Consequently it was decided that such a thesis was not only justifiable, but would have significant practical value as well.



## II. A DESCRIPTION OF THE INTERFACE

In order to understand how the INTERFACE was accomplished, it is necessary to know a little about the sequence of events in the original SIMUL8S package. All events were controlled by one routine called MAIN. First assembly was accomplished and the resulting object code stored in an 8K array called IADD. All positions in IADD not filled with object code were initialized to zero. After assembly there was an option to load additional object code directly into IADD bypassing the assembly step. This option is primarily used to load the FLOATING POINT PACKAGE.<sup>5</sup> Next there was an option to create a paper tape image of IADD on magnetic tape. This paper tape image was a listing of all filled IADD locations in the proper format for later input to the PDP-8 computer via paper tape. Finally there was an option to simulate execution of the object code residing in IADD.

### A. INTERFACING THE COMPILER AND SIMUL8S

It was determined that the best way to simulate the object code generated by the compiler was to read it into IADD and then execute the simulation option in the normal manner. This could not be done directly, however, because the object code generated by FORTRAN/8 was in a different format from that generated by the assembler and stored into IADD. The FORTRAN/8 object code was originally intended to be written onto seven track magnetic tape.

---

<sup>5</sup>The name FLOATING POINT PACKAGE refers to any one of the four floating-point software programs available from Digital Equipment Corporation [Ref. 3]. Unless otherwise specified, mention in this thesis of the term FLOATING POINT PACKAGE refers to the basic floating-point package DEC-08-YQ1A-PB.



The IBM system 360 is a 32 bit word machine with eight bits per byte. When writing an eight bit byte onto a seven track magnetic tape the upper two bits in the byte are lost. Hence FORTRAN/8 produced object code wherein each byte was padded with two zeros in the upper two bits. This format was acceptable for seven track magnetic tape, but not for input to IADD.

Words in IADD have information only in the lower three bytes with the upper byte all zeros. Hence each word of FORTRAN/8 object code was transformed into proper format by stripping each byte of its two zeros and placing the resulting eight zeros, from each four bytes, at the beginning of the word. Once the transformation was completed the word was read into IADD, and when all of the FORTRAN/8 object code had been transformed and read into IADD, simulation was carried out normally.

As a result of the INTERFACE the original SIMUL8S sequence of events was altered. The first option the user now has is the reading of the FORTRAN/8 object module into IADD. If this first option is not elected then the rest of the event sequence is identical to the original one. If this first option is taken then provision is made for optional assembly. If the assembly option is taken the original event sequence continues beginning with assembly. If not taken then the original event sequence continues beginning with the option to create a paper tape image of IADD on magnetic tape.

There are two basic paths the user may follow in executing the INTERFACE. First there is assembly followed by simulated execution of the resulting object code. Once the assembled code has been debugged satisfactorily the user may create a paper tape image on magnetic tape





to be later input to the PDP-8 for actual execution. The second path is similar to the first except that the user substitutes compilation for assembly.

As an aid in debugging during simulation, a complete dump of memory (IADD) can be very helpful. This can only be specified during assembly. Hence execution of the second path above would seem to preclude the possibility of obtaining a memory dump. To circumvent this problem a third path can be followed.<sup>6</sup> In this case compilation is followed by a trivial assembly, wherein the dump is specified. During simulation, when the resulting object code is executed, the dump occurs.

#### B. INTERFACING THE OBJECT MODULE AND SIMUL8S

Originally SIMUL8S dumped its object module on nine track magnetic tape. The object module was in proper format for input to the PDP-8 via the BINARY LOADER.<sup>7</sup> Paper tape is the only common input medium acceptable to both the PDP-8/S and the PDP-8/E. Hence for the original object module to work effectively it would have to be transferred from nine track magnetic tape to paper tape. At present the IBM System 360 at the Naval Postgraduate School does not have this capability. This is because all input to the ASR 33 teletype must pass through the 2702 control unit. The control unit punches the eighth position in each and every frame of paper tape and this is unacceptable to the BINARY LOADER.

---

<sup>6</sup>Further explanation of this path is given in Appendix A.

<sup>7</sup>The BINARY LOADER [Ref. 4] is a short routine for reading and storing information contained in binary-coded tapes and uses the 33-ASR reader or the High-Speed Reader.



The problem now was to determine how an acceptable paper tape could be punched. The SDS 9300 Computer System at the Naval Postgraduate School was considered next.<sup>8</sup> Although this system has a paper tape punch, no hardware exists for punching in the eighth position as is required by the BINARY LOADER. It was finally discovered that the CDC-160 Computer System at the Naval Postgraduate School (Spanagel Hall) had the capability of punching an acceptable paper tape.<sup>9</sup> At the time of discovery this system was using seven-eighths inch paper tape which was not wide enough for punches in the eighth position. Recently, however, the system was switched over to one inch paper tape which allows punching of the eighth position.

The most effective method of transferring information from the IBM System 360 to the CDC-160 Computer System is via seven track magnetic tape written at 200 bits per inch in odd parity. Since the IBM System 360 has the capability of writing seven track magnetic tape, the next step in the INTERFACE was to dump the SIMUL8S object module on this tape and then carry the tape over to the CDC-160 Computer System for further processing.

---

<sup>8</sup> The SDS 9300 is a high-speed, general purpose, digital computer manufactured by Scientific Data Systems, El Segundo, California.

<sup>9</sup> The CDC-160 is a 4K, parallel, single address electronic data processor manufactured by Control Data Corporation, Minneapolis, Minnesota.



Going from the IBM System 360 to seven track magnetic tape generated the same problem, but in reverse, that was encountered in going from FORTRAN/8 to the IBM System 360. In other words each byte in each word of the object module had to be padded with zeros in the upper two bits so that the writing onto seven track magnetic tape would not cause any loss of information. Before this could be accomplished an IBM System 360 assembly language program allowing bit shifting to the left had to be written. It was called ISHFTL and was similar in nature to the right shifting assembly language program found in SIMUL8S called ISHIFT.

Each word in IADD is transformed as specified above before writing onto seven track magnetic tape. The entire contents of IADD are dumped to tape including locations filled with zeros. A total of eight records, each 1024 words in length, are written with an end of file mark following the last record. This, then, is how the paper tape image discussed in the original sequence of events [Sect. II. A] is written.

### C. INTERFACING THE OBJECT MODULE AND PDP-8

The most efficient method of paper tape input to the PDP-8 is via the BINARY LOADER. To be acceptable to the BINARY LOADER a paper tape must be in BINARY format.<sup>10</sup> Thus the final step in the INTERFACE was to produce a paper tape in BINARY format which represented the object module dumped to seven track magnetic tape by SIMUL8S.

---

<sup>10</sup>A good explanation of BINARY format can be found on page 4-15 of [Ref. 5].



To accomplish this a program was written in CDC-160 assembly language [Ref.6]. This program runs on the CDC-160, accepts seven track magnetic tape written by SIMUL8S for input, and produces as output a BINARY paper tape acceptable to the PDP-8 BINARY LOADER. The general flow of the program is as follows: Punch the leader; sequentially read and punch each of the eight magnetic tape records in proper BINARY format; punch two zero frames to represent the checksum; punch the trailer; and rewind the magnetic tape to the unload point. If at any time an input parity error is sensed, the magnetic tape is rewound to the load point so that the program can be executed again. It was decided to dispense with the writing of the proper checksum due to the differences between the PDP-8 and the CDC-160. The former is a two's complement machine while the latter is a one's complement machine. This will not in any way interfere with the proper loading of the object module into the PDP-8.





### III. INTERFACE TEST RESULTS

In light of the discussion which follows a description of job execution steps will be helpful. For assembly and subsequent simulation the SIMUL8S package is executed specifying the desired options. This is also the case for each of the following: Assembly and the subsequent creation of a paper tape image; input of FORTRAN/8 object code and subsequent simulation or creation of a paper tape image; and combination of assembly and input of FORTRAN/8 object code followed by simulation or creation of a paper tape image. The execution of FORTRAN/8 to generate compiled object code requires a separate job step.

#### A. SIMULATED EXECUTION OF FORTRAN/8 OBJECT MODULE

Two FORTRAN/8 compilations were tested. The first program simply read one real value from the teletype and then printed it back out on the teletype. This program was simulated successfully.

The second program solved for the two roots of a quadratic equation using the quadratic formula. The formula was modified in that no square root was calculated since the FORTRAN/8 compiler would not properly compile the square root statement. In order to still provide accurate answers, the input values used for A, B, and C were such that the discriminant evaluated to one. The two following sets of input values were tested: 3/7/4 and 2/5/3. The results of the simulation were as follows: For 3/7/4, +46 and -60.6; for 2/5/3 +22.6 and -32.6. The results should have been as follows: For 3/7/4, -1.33 and -1.0; for 2/5/3, -1 and -1.5. Discussion of possible causes of the incorrect answers is covered in section IV.



For comparison, an assembled version of the quadratic formula was simulated using the same input data. The results were as follows: For 3/7/4, -1.66 and -.602; for 2/5/3, -1.86 and -.602. This rather disturbing result is also discussed in section IV.

#### B. PDP-8/S OBJECT MODULE EXECUTION

Three object module paper tape images were written onto seven track magnetic tape by SIMUL8S. The tape was used as input to the CDC-160 program and three paper tapes in BINARY format were produced. These paper tapes were individually tested by reading them into the PDP-8/S via the BINARY LOADER and then executing the resulting object module.

##### 1. Compiled Module

The compiled quadratic formula object module was tested with the following results: For 3/7/4, +48.0 and -50.3; for 2/5/3, +24.0 and -26.5.

##### 2. Assembled Modules

First the assembled quadratic formula was tested on the input values of 3/7/4 and 2/5/3. The results were the correct answers. For the second test the routines found on page 5-11 of [Ref. 5] which accept, store, and print teletype characters were used. This object module also executed correctly.

#### C. PDP-8/E OBJECT MODULE EXECUTION

Four object module paper tape images were produced and tested in the same manner as described for the PDP-8/S.



## 1. Compiled Modules

The compiled quadratic formula object module was tested with the same results as those for the PDP-8/S. In addition the object module produced by the simple read-and-write-one-variable program was tested successfully.

## 2. Assembled Modules

The same two object modules used for the PDP-8/S test were executed with results identical to those achieved on the PDP-8/S.



#### IV. PROBLEM AREAS

After the purchase of SIMUL8S the first task was to check it out to ensure proper operation. During this phase of program checkout, several problems were uncovered and these, along with those uncovered in FORTRAN/8 will be discussed in this section.

##### A. RESOLVED

There were several problems associated with SIMUL8S which have been corrected. A discussion of these follows.

##### 1. Assembler

The SIMUL8S USER GUIDE specified that in order to terminate the input of an object module that bypassed the assembly step, a card with a \$ punched in column one should be used. It was found, however, that the use of a \$ was incorrect and that only the use of a card with STOP punched beginning in column one would allow proper execution. This error is documented in Appendix E.

During the SIMUL8S program checkout phase, incorrect assembly source input was tested. The assembler correctly diagnosed the error; however, the error message printed was garbled. The cause was isolated to a format statement in MAIN and subsequently corrected. In addition, upon completion of assembly of the incorrect source input, program termination did not occur as specified by the SIMUL8S documentation. The problem was again traced to a faulty statement in MAIN and subsequently corrected.





## 2. Simulator

Before the assembled object code for the quadratic formula is simulated, prior loading of the FLOATING POINT PACKAGE object module must have been accomplished. During initial simulation of the assembled quadratic formula, the simulator abnormally terminated after finding a machine language instruction which it could not recognize. This instruction was found to belong to the FLOATING POINT PACKAGE. This rather alarming development meant that SIMUL8S could not be used to simulate any program that needed the FLOATING POINT PACKAGE. Since this package is used whenever real numbers are involved, it was necessary to modify the simulator so that it would accept the unrecognized instruction.

To prevent reoccurrence of this problem, all four possible FLOATING POINT PACKAGES were searched for instructions that the simulator would fail to recognize. The five following instructions were found: CLL IAC, SZL CLA, CLA CML, CMA IAC CML, and SMA SZA CLA. The simulator was appropriately modified to accept the instructions and then tested with the assembled quadratic formula. As stated in Section III. A., simulation ran to completion, but incorrect results were obtained. However, as documented in Section III. B.2., proper execution of the assembled quadratic formula was observed in the PDP-8. This indicates that errors exist in the SIMUL8S simulation of the FLOATING POINT PACKAGE. It is felt that the responsibility for correction lies with the software manufacturer.



## B. UNRESOLVED

Several problems were uncovered during the testing of the INTERFACE which will be documented in this section. The problem in the SIMUL8S simulation of the FLOATING POINT PACKAGE has already be discussed. The remainder of the unresolved problems pertain to FORTRAN/8.

### 1. Compiler

Neither the PDP-8/S nor the PDP-8/E has an Extended Arithmetic Element (EAE).<sup>11</sup> This means that the machine language instructions associated with the EAE are not acceptable to either of the PDP-8 machines currently assigned to the Naval Postgraduate School. The FORTRAN/8 compiler frequently produces these unacceptable instructions. There are eight instructions in all and they are called MQ MICROINSTRUCTIONS. Some of these instructions are generated in at least two known cases as follows: The writing of integers on the teletype and the conversion of integers to real numbers.

Documentation on FORTRAN/8 states that all input is assumed to be via teletype. However, FORTRAN/8 fails to generate the required code needed for teletype input initialization. Both compiled object modules that were tested on the PDP-8 failed to allow teletype input.<sup>12</sup> However, manual loading on the PDP-8 of the instructions 6032 (KCC) and 6046 (TLS) allowed execution of the test programs. These two instructions were loaded into the octal locations 176 and 177. Execution was then begun from 176.

---

<sup>11</sup> The EAE is a PDP-8 hardware option which provides circuitry to perform arithmetic operations which can not be directly performed with the basic PDP-8 instruction set.

<sup>12</sup> Simulated execution of the incorrect code is permitted, however, as the simulator provides for the necessary teletype input initialization.



In general, programs run on the PDP-8 should leave the last page in each field of the memory of the PDP-8 empty. This allows permanent residence of the BINARY LOADER in this last page. FORTRAN/8 loads several subprograms into this last page. As a result, improper loading of the FORTRAN/8 object code into the PDP-8 memory can occur. This happens whenever object code is loaded into the same field that the BINARY LOADER resides. In effect the object code tries to overlay the BINARY LOADER causing termination of the loading process. On the PDP-8/E this problem can be avoided since there are two fields of memory totaling 8K. In this case the BINARY LOADER usually resides in field one, while the object code is loaded into field zero. The PDP-8/S, however, has only 4K of memory; and thus only one memory field exists. It is therefore impossible to obtain proper loading of the FORTRAN/8 object code into the PDP-8/S.

When the compiled quadratic formula object module was tested on the PDP-8, incorrect results were obtained. This indicates that FORTRAN/8 has errors in its object code generation in addition to the ones already discussed. This conclusion was reached in view of the fact that the assembled quadratic formula object code was tested satisfactorily on the PDP-8. Mention is also made of the simple read-and-write-one-variable program which was tested satisfactorily on the PDP-8.

During compilation of the quadratic formula the square root statement would not compile properly. The error message received was: "\*\*\*ERROR, SQRT REQUIRES REAL EXPRESSION\*\*\*." This expression was of type real, having been so declared at the beginning of the program.



## V. CHANGES MADE TO THE SIMUL8S PACKAGE

In this section the changes made to the original SIMUL8S package will be documented.

### A. GENERAL CHANGES

Features added to or deleted from the SIMUL8S package will be discussed next.

#### 1. PDP-8/E Simulation

The original SIMUL8S made no provision for simulation of the PDP-8/E. The tracing debug feature of the simulator made this a definite limitation to the PDP-8/E user group. Consequently, modifications were made to SIMUL8S in order to allow PDP-8/E simulation.

#### 2. Teletype Input

Originally only 500 teletype characters were allowed as input to the simulator. SIMUL8S has been modified so that 1000 characters are now allowed.

#### 3. EAE Simulation

Simulation of the EAE has been deleted from SIMUL8S. This was done to prevent the user from successfully simulating object code containing EAE instructions only to discover later that his program would not execute properly on the PDP-8. In addition, a reduction in the region size required to execute SIMUL8S was realized, thereby reducing the user's job turnaround time. EAE instructions are still recognized by the simulation, but an appropriate error message is generated and simulation is terminated. Assembly of EAE instructions is allowed.





Provision has been made for restoration of EAE simulation in case that option is desired.

#### B. SPECIFIC SUBROUTINE CHANGES

All subroutines having references to any of the vectors which follow were changed to permit PDP-8/E simulation: GRPTIM, REGTIM, FINDIR, AUTOIN, and TIMTAB. Also all subroutines having references to the vector ITABK were changed to permit a larger teletype input to the simulator. The subroutine XARITH which simulated the EAE was deleted. The subroutines LEADER and TAPPAK which were used in the writing of the original paper tape image were deleted. The subroutine MAIN was altered to bring about the new sequence of events specified in the description of the INTERFACE. The subroutines DOUBLE and FIND were altered to allow simulation of the five unrecognized FLOATING POINT PACKAGE instructions, and to generate the error message from attempted simulation of an EAE instruction.



## VI. RECOMMENDATIONS

This section describes areas where more work needs to be done, and gives a few helpful recommendations to the PDP-8 user groups.

### A. ZEROING OF PDP-8 MEMORY

The program on the CDC-160 which produces paper tape is designed to produce as short a tape as possible. This is accomplished in the following manner. Whenever a string of greater than two zeros is read, only the first two are punched on paper tape. Before the next non-zero piece of data is punched the current value of the program counter is punched. This means that if the PDP-8 user has a string of memory locations he wants initialized to zero, he must zero them himself. An easier method of accomplishing this is to zero the PDP-8 memory before loading anything. The routine which follows may be used to zero one field of memory of the PDP-8/E. Slight modification may be necessary for use on the PDP-8/S.

| <u>ADDRESS</u> | <u>CODE</u> | <u>LABEL</u> | <u>MNEMONIC DESCRIPTION</u> |
|----------------|-------------|--------------|-----------------------------|
| 0000           | 7300        |              | CLA CLL                     |
| 0001           | 3405        |              | DCA I A                     |
| 0002           | 2005        |              | ISZ A                       |
| 0003           | 5001        |              | JMP .-2                     |
| 0004           | 7402        |              | HLT                         |
| 0005           | 0000        | A,           | 0                           |

### B. COMPILER CHANGES

In support of the discussion on FORTRAN/8 problem areas, the following suggestions are offered.

Wherever incorrect MQ MICROINSTRUCTIONS are generated, a different code generation that will effect the same result must be accomplished.



In order to allow correct teletype input initialization, the two instructions mentioned earlier could be loaded beginning at octal location 200 prior to the loading of subsequent object code. This solution would probably require some modification of the subsequent object code.

In order to keep the last memory page empty the subprograms EXPONENTIATION, DIVIDE, MULTIPLY, and SUBSCRIPT must be moved elsewhere. A good possibility would be from octal location 5473 downward.

### C. PI-1250-1 INTERFACE

Paper tape input to the PDP-8/S via the slow speed teletype reader is a very time consuming process. The PI-1250-1 Data Handling System is a seven track magnetic tape drive compatible with IBM equipments. Hence it should be feasible to build an interface so that the paper tape image written on seven track magnetic tape by SIMUL8S could be used as input to the PDP-8/S via the PI-1250-1 Data Handling System.<sup>13</sup>

---

<sup>13</sup>Reference 7 describes an interactive tape driving routine which operates in conjunction with a teletype and the PI-1250-1 Data Handling System. This reference is suggested as an aid in the building of the interface discussed above.



#### D. HAND PLACEMENT OF CHECKSUM

The normal operation of the BINARY LOADER is to calculate a checksum as it loads a paper tape into the PDP-8. The value is compared to the checksum punched on the tape by subtracting one from the other. If the two agree a zero is left in the accumulator. If they do not agree then the result of the subtraction is left in the accumulator. Since zero is punched for the checksum by the INTERFACE, a non-zero number will be left in the accumulator upon completion of loading. If the user writes this number from the accumulator onto the paper tape the first time he loads it, then subsequent loadings can be checked to see if the same number is generated in the accumulator. As long as the number agrees with that on the tape, the tape has been read correctly. This method has been tested successfully on both the PDP-8/S and the PDP-8/E.





## VII. CONCLUSIONS

Successful completion of the INTERFACE has provided the PDP-8 user community with an unlimited number of virtual PDP-8 computers. In addition, the use of a large computer system, such as the IBM System 360, provides an operating system to handle the housekeeping duties normally required of the PDP-8 user.

The present configuration of SIMUL8S requires an IBM System 360 region size of 180K. Execution times have ranged between three seconds and four minutes depending upon options selected. The writing of the paper tape image on seven track magnetic tape when no simulation is desired takes about three seconds. Execution of FORTRAN/8 requires 150K and averages three seconds. Approximately ten minutes is required for the conversion of the paper tape image on magnetic tape to the actual paper tape. This time includes set up and take down time as well as actual program execution on the CDC-160.

The FORTRAN/8 problem areas discussed in an earlier section seem reasonable for student projects in the compiler construction course (CS 4113). The amount and type of work required is considered to be consistent with the objectives of that course.



## APPENDIX A

### INTERFACE USER MANUAL

#### EXECUTION OF SIMUL8S

Proficiency in the use of SIMUL8S for assembly, simulation, and generation of the paper tape image can only be obtained after the PDP-8 user has become thoroughly familiar with the SIMUL8S user manuals listed in references 1 and 2. The one technique not documented in these references is the combination of assembly and FORTRAN/8 object code input for the purpose of obtaining a memory dump during simulation.

This technique is described as follows:

1. Execute the FORTRAN/8 compilation of the desired program, leaving out the STOP CARD.<sup>14</sup> Be sure to include the toggle, \$T, for a memory map. (An error will occur due to the deleted STOP card, but it will only affect the generation of the HLT instruction-which is what is desired.)
2. With the output of step one in hand, scan down the memory map beginning at location 200 until the last program-generated instruction is found. Make a note of the address of the next sequential location.
3. Execute SIMUL8S with the combination option using a trivial assembly language program such as the following:

\*ADDRESS NOTED ABOVE

CLA

\$DUMP2

HLT

4. When the object module is simulated, execution will eventually reach the address noted above. At this point execution of the assembled code will begin and the memory dump will occur.

---

<sup>14</sup>

The STOP card referenced is the one at the end of the main program immediately preceding the END card. Any subroutines included must precede the main program (allowable in FORTRAN/8).



## EXECUTION OF FORTAN/8

Familiarity with the following areas in reference 8 will aid the PDP-8 user in compiling FORTRAN/8 source code properly:

| <u>AREA</u>                     | <u>PAGES</u> |
|---------------------------------|--------------|
| BNF FOR FORTRAN/8               | 47-50        |
| STATEMENTS ALLOWED IN FORTRAN/8 | 51-54        |
| FORTAN/8 LISTING CONTROLS       | 114-115      |

## EXECUTION OF THE CDC-160 PAPER TAPE CONVERSION PROGRAM

The CDC-160 computer is located in room 521 of Spanagel Hall. The seven track magnetic tape generated on the IBM System 360 must be carried to the CDC-160 and placed on the associated magnetic tape drive. The basic procedures for turning power on and off, operating the tape drive, and operating the computer are listed in a user manual kept on the top surface of the computer. The sequence of events required to produce the paper tape for later input to the PDP-8 is as follows:

1. Power on the CDC-160 computer and associated tape drive.  
Connect tape drive cables (located behind plotter).
2. Load the seven track magnetic tape obtained from the IBM System 360 on the tape drive. Press the forward tape control button for several seconds. Press CLEAR. Press REWIND LOAD.
3. Turn on the CDC-160 paper tape reader.
4. Take the paper tape found in the plastic drawer marked PDP-8 PROGRAM and place it properly into the reader.
5. Zero the CDC-160 memory.
6. Load the PDP-8 PROGRAM tape into the CDC-160 beginning at location zero.
7. Check to see that the A register contains the proper checksum of 2703.
8. Turn off the paper tape reader.
9. Turn on the paper tape punch.



10. Execute the PDP-8 PROGRAM from location zero. If a parity error occurs the magnetic tape will be rewound to the load point, and 0077 will remain in the Z register. Execution from zero can be attempted again. If a parity error reoccurs stop all execution; go back to the IBM System 360 and recreate the magnetic tape.
11. Upon normal completion of execution the magnetic tape will be rewound to the unload point, and 7777 will remain in the Z register.
12. Halt the CDC-160. Play out the punched paper tape until it can be torn off.
13. Turn off the paper tape punch.
14. Unload the magnetic tape from the tape drive.
15. Power off the CDC-160 computer and associated tape drive. Disconnect the cables.
16. Put away the PDP-8 PROGRAM paper tape.

In the event the user is interested in obtaining a paper tape of another file on the magnetic tape, the PDP-8 PROGRAM can be so modified. This is done once the program is already loaded into the CDC-160 memory. To effect this modification load into location 0112 the instruction 7700(halt). This will cause the program to process one file on the magnetic tape and halt without rewinding to the unload point. In this manner the user can sequence through the magnetic tape until the desired file is located. It should be noted that during the sequencing a paper tape will be produced for each file processed. The program can be restored to normal operation by reloading 0112 with the instruction 6102. The manner in which the extra files are created on the seven track magnetic tape is documented in Appendix B.





## APPENDIX B

### JOB CONTROL LANGUAGE

It is expected that as the demand for use of this INTERFACE increases, the JOB CONTROL LANGUAGE (JCL) necessary for proper execution of SIMUL8S and FORTRAN/8 will be placed in a catalogued procedure. The following JCL was used for execution of SIMUL8S and FORTRAN/8, and was valid at the time of this writing.

#### FOR SIMUL8S

```
//KILLER08 JOB (2063,0622XT,CS12),'BARNES'
//JOB LIB DD DSN=S2063.BARLIB,DISP=(OLD,PASS),VOL=SER=CELO03,UNIT=2321
//GO EXEC PGM=ASIM3,REGION=180K
1 //GO.FT06F001 DD SYSOUT=A,SPACE=(CYL,(3,1))
* //GO.FT08F001 DD DSN=S2063.IRK.FILE1,UNIT=2314,VOL=SER=DUFFY,DISP=OLD,
// LABEL=(,,IN)
//GO.FT10F001 DD UNIT=SYSDA,DSN=&TEMP1,DISP=(NEW,DELETE),
2 // DCB=(RECFM=VS,LRECL=292,BLKSIZE=296),SPACE=(CYL,(3,1))
* //GO.FT11F001 DD UNIT=2400-1,VOL=SER=BARNES,DISP=(NEW,PASS),
// DCB=(DEN=0,RECFM=F,LRECL=2048,BLKSIZE=2048),LABEL=(1,BLP)
3 * //FT05F001 DD *
//
```

\*<sup>1</sup> This card allows input of FORTRAN/8 object code to SIMUL8S. Leave it out unless this option is desired.

\*<sup>2</sup> These two cards allow the writing of the paper tape image on seven track magentic tape. Leave them out unless this option is desired. By modifying the parameter LABEL=(1, BLP) subsequent files can be written on the same magnetic tape. This is done by changing the 1 to a 2 for the second file, the 2 to a 3 for the third file, etc.

\*<sup>3</sup> Data cards for the execution of SIMUL8S follow this card.



## FOR FORTRAN/8

```
//KILLER07 JOB (2063,0622XT,CS12),'BARNES',TIME=(,5)
//JOB LIB DD DSN=SYS3.XPL.MONITOR,UNIT=2314,VOL=SER=LINDA,
// DISP=(SHR,PASS)
//GO EXEC PGM=XPLSM,REGION=150K,COND=(0,NE)
//PROGRAM DD DISP=(OLD,KEEP),UNIT=2314,DSNAME=F2853.TUCKER.FORTRAN8,
//      VOLUME=SER=LINDA
//FILE1 DD VOL=SER=DUFFY,UNIT=2314,DISP=(OLD,KEEP),
//      SPACE=(TRK,4,RLSE),DCB=(RECFM=F,BLKSIZE=3600),
//      LABEL=RETPD=180,DSN=S2063.IRK.FILE1
//SYS PRINT DD SYSOUT=A,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=133),
*1 //      SPACE=(CYL,(2,1))
//SYS IN DD *
//
```

\* The FORTRAN/8 input source deck follows this card. The last card in the input deck must be \$GO, punched beginning in column one.

If the necessary JCL has not been catalogued, it is suggested that the user check the validity of the above JCL before using it. The Duty Consultant found in INGERSOLL 146 should be able to help the user in this regard.



CDC-160 PROGRAM LISTING

program to punch paper tape in binary format  
for pdp-8 input. run on cdc-160 using ibm-360/67  
mag tape output for input.

32



|      |      |       |     |       |                                |
|------|------|-------|-----|-------|--------------------------------|
| 0027 | 6102 |       | nzf | 2     | (first word address leader)    |
| 0030 | 7001 | fvald |     | 7001  | (tape unit ready test:         |
| 0031 | 7500 | turt  | exf |       | request status, if not         |
| 0032 | 1141 |       |     | 1141  | ready repeat)                  |
| 0033 | 7600 |       | ina |       |                                |
| 0034 | 0202 |       | lpn | 2     | (read one record:              |
| 0035 | 6504 |       | nzb | turt  | input one mag tape record to   |
| 0036 | 7500 | ror   | exf |       | memory locations 1000 to 1777) |
| 0037 | 2131 |       | inp | 2131  |                                |
| 0040 | 7203 |       |     | fwamc |                                |
| 0041 | 3000 |       |     | 3000  |                                |
| 0042 | 6102 |       | nzf | 2     | (first word add mem counter)   |
| 0043 | 1000 | fwamc |     | 1000  | (second ready test:            |
| 0044 | 7500 | srt   | exf |       | request status and store in    |
| 0045 | 1141 |       |     | 1141  | cond)                          |
| 0046 | 7600 |       | ina |       |                                |
| 0047 | 4212 |       | stf | cond  |                                |
| 0050 | 0202 |       | lpn | 2     | (end of file test:             |
| 0051 | 6505 |       | nzb | 5     | if eof was read jump to pt.    |
| 0052 | 2207 | eoft  | ldf | cond  | if a parity error occurred     |
| 0053 | 0220 |       | lpn | 20    | jump to pe.                    |
| 0054 | 6130 |       | nzf | pt    | ow jump to mdpbf)              |
| 0055 | 2204 |       | ldf | cond  |                                |
| 0056 | 0204 |       | lpn | 4     |                                |
| 0057 | 6121 |       | nzf | pe    |                                |
| 0060 | 6040 |       | zjf | mdpbf |                                |
| 0061 | 0000 | cond  |     | 0     | (condition)                    |
| 0062 | 0000 |       |     | 0     |                                |
| 0063 | 0000 |       |     | 0     |                                |
| 0064 | 0000 |       |     | 0     |                                |
| 0065 | 0000 |       |     | 0     |                                |
| 0066 | 0000 |       |     | 0     |                                |
| 0067 | 0000 |       |     | 0     |                                |
| 0070 | 0000 |       |     | 0     | (if program is loaded          |
| 0071 | 0000 |       |     | 0     | at 0000 these next seven       |
| 0072 | 0000 |       |     | 0     | locations are used for         |





|      |      |       |        |                               |
|------|------|-------|--------|-------------------------------|
| 0073 | 0000 |       | 0      | temporary storage)            |
| 0074 | 0000 |       | 0      |                               |
| 0075 | 0000 |       | 0      |                               |
| 0076 | 0000 |       | 0      |                               |
| 0077 | 0000 |       | 0      |                               |
| 0100 | 7500 | pe    |        | (parity error:                |
| 0101 | 1161 |       | 1161   | rewind tape to load point.    |
| 0102 | 0400 | ldn   |        | load 0 into accumulator.      |
| 0103 | 0077 |       | 77     | halt with error code 0077)    |
| 0104 | 7500 | pt    |        | (punch trailer:               |
| 0105 | 4104 | exf   | 4104   | first output two frames       |
| 0106 | 7400 | otn   |        | of zero for the check sum,    |
| 0107 | 7400 | otn   |        | then punch the trailer)       |
| 0110 | 7303 | out   |        |                               |
| 0111 | 7145 |       | fwatr  |                               |
| 0112 | 6102 |       | 7145   |                               |
| 0113 | 7001 | nzf   | 2      |                               |
| 0114 | 7500 |       | 7001   | (first word address trailer)  |
| 0115 | 1151 | exf   |        | (rewind tape to unload point. |
| 0116 | 0400 |       | 1151   | load 0 into accumulator.      |
| 0117 | 7777 | ldn   | 7777   | halt with normal completion   |
| 0120 | 0400 | ldn   |        | code 7777)                    |
| 0121 | 4244 | mdpbf |        | (manipulate data, punch in    |
| 0122 | 4244 | stf   |        | binary form:                  |
| 0123 | 4244 | stf   | switch | set switch, tag, and          |
| 0124 | 2215 | stf   | tag    | flag to zero)                 |
| 0125 | 4070 | ldf   | flag   | (load mcs)                    |
| 0126 | 2214 | std   | mcs    | (store in 0070)               |
| 0127 | 4071 | ldf   | 70     | (load 1s)                     |
| 0130 | 2213 | std   | 1s     | (store in 0071)               |
| 0131 | 4074 | ldf   | 71.    | (load b1)                     |
| 0132 | 2212 | std   | b1     | (store in 0074)               |
| 0133 | 4075 | ldf   | 74     | (load b2)                     |
| 0134 | 0101 | std   | b2     | (store in 0075)               |
| 0135 | 3200 | pta   | 75     |                               |
| 0136 | 0114 | adf   |        | (store address of eb in 0076) |



|      |      |        |     |        |                               |
|------|------|--------|-----|--------|-------------------------------|
| 0137 | 4076 |        | std | 76     | (go to loopl)                 |
| 0140 | 6105 |        | nzf | loopl  | (memory counter start:        |
| 0141 | 1000 | mcs    |     | 1000   | for mag tape input)           |
| 0142 | 3000 | ls     |     | 3000   | (1 start: for punch output)   |
| 0143 | 7777 | b1     |     | 7777   | (new setting for pc)          |
| 0144 | 0000 | b2     |     | 0      | (new setting for pc)          |
| 0145 | 2170 | loopl  | ldi | 70     | (load one piece of data       |
|      |      |        |     |        | from mag tape input area.     |
|      |      |        |     |        | if data is zero go to zd)     |
| 0146 | 6027 |        | zjf | zd     | (set switch to zero)          |
| 0147 | 0400 |        | ldn |        | (test tag:                    |
| 0150 | 4215 |        | stf | switch | if zero go to nla)            |
| 0151 | 2215 | tt     | ldf | tag    | (load sequential data:        |
| 0152 | 6030 |        | zjf | nla    | reload data from mag tape     |
| 0153 | 2170 | lsd    | ldi | 70     | input.                        |
| 0154 | 1214 |        | lpf | lm     | store upper half in output    |
| 0155 | 0111 |        | ls6 |        | area. increment 1 by 1.       |
| 0156 | 4171 |        | sti | 71     | reload data from mag tape     |
| 0157 | 5471 |        | aod | 71     | input.                        |
| 0160 | 2170 |        | ldi | 70     | store lower half in output    |
| 0161 | 0277 |        | lpn | 77     | area. increment 1 by 1.       |
| 0162 | 4171 |        | sti | 71     | go to pczt)                   |
| 0163 | 5471 |        | aod | 71     | (used to avoid punching       |
| 0164 | 6136 |        | nzf | pczt   | series of zeroes in the       |
| 0165 | 0000 | switch |     | 0      | mag tape input data)          |
|      |      |        |     |        | when zero, causes value of    |
| 0166 | 0000 | tag    |     | 0      | pc to be punched)             |
| 0167 | 0000 | flag   |     | 0      | (set to zero when pc is zero) |
| 0170 | 7700 | lm     |     | 7700   | (lower mask)                  |
| 0171 | 2070 | eidt   | ldd | 70     | (end input data test: load    |
| 0172 | 3730 |        | sbb | ls     | current value of mcs.         |
| 0173 | 6065 |        | zjf | pd     | subtract 3000. if zero go     |
| 0174 | 6527 |        | nzb | loopl  | to pd. ow go to loopl)        |
| 0175 | 2310 | zd     | ldb | switch | (zero data: load switch.      |



|      |      |      |     |        |                              |
|------|------|------|-----|--------|------------------------------|
| 0176 | 6122 |      | nzf | id     | if not zero go to id.        |
| 0177 | 0401 |      | ldn | 1      |                              |
| 0200 | 4313 |      | stb | switch | ow set switch to 1.          |
| 0201 | 6530 |      | nzb | tt     | go to tt)                    |
| 0202 | 2072 | nla  | ldd | 72     | (new load address: load      |
| 0203 | 1313 |      | lpb | lm     | current value of pc. save    |
| 0204 | 0111 |      | ls6 |        | upper half.                  |
| 0205 | 3212 |      | adf | sls    | add sls to upper half.       |
| 0206 | 4171 |      | sti | 71     | store result in output area. |
| 0207 | 5471 |      | aod | 71     | increment 1 by 1.            |
| 0210 | 2072 |      | ldd | 72     | reload pc.                   |
| 0211 | 0277 |      | lpn | 77     |                              |
| 0212 | 4171 |      | sti | 71     | store lower half in output   |
| 0213 | 5471 |      | aod | 71     | area. increment 1 by 1.      |
| 0214 | 0401 |      | ldn | 1      |                              |
| 0215 | 4327 |      | stb | tag    | set tag to 1.                |
| 0216 | 6543 |      | nzb | lsd    | go to lsd)                   |
| 0217 | 0100 | sls  |     | 100    | (start location signal)      |
| 0220 | 0400 | id   | ldn |        | (ignore data:                |
| 0221 | 4333 |      | stb | tag    | set tag to zero)             |
| 0222 | 2333 | pczt | ldb | flag   | (program counter zero test:  |
| 0223 | 6020 |      | zjf | sf     | load flag. if zero go to sf) |
| 0224 | 2072 | bt   | ldd | 72     | (bump test: load pc.         |
| 0225 | 3621 |      | sbf | abl    | subtract 7776.               |
| 0226 | 6006 |      | zjf | sbl    | if zero go to sbl.           |
| 0227 | 2072 |      | ldd | 72     | reload pc.                   |
| 0230 | 3617 |      | sbf | ab2    | subtract 7777.               |
| 0231 | 6006 |      | zjf | sb2    | if zero go to sb2)           |
| 0232 | 5472 | rb   | aod | 72     | (regular bump: increment pc  |
| 0233 | 7076 |      | jpi | 76     | by 1. go to eb)              |
| 0234 | 2074 | sbl  | ldd | 74     | (simulate bump1: load 7777.  |
| 0235 | 4072 |      | std | 72     | set pc to 7777.              |
| 0236 | 6112 |      | nzf | eb     | go to eb)                    |
| 0237 | 2075 | sb2  | ldd | 75     | (simulate bump2: load zero.  |
| 0240 | 4072 |      | std | 72     | set pc to 0000.              |
| 0241 | 4352 |      | stb | flag   | set flag to zero.            |



|      |      |        |     |        |   |
|------|------|--------|-----|--------|---|
| 0242 | 6006 |        | zjf | eb     | go to eb)                                 |
| 0243 | 0401 | sf     | ldn | 1      | (set flag:                                |
| 0244 | 4355 |        | stb | flag   | set flag to 1.                            |
| 0245 | 6513 |        | nzb | rb     | go to rb)                                 |
| 0246 | 7776 | abl    |     | 7776   | (avoid bump1)                             |
| 0247 | 7777 | ab2    |     | 7777   |   |
| 0250 | 5470 | eb     | aod | 70     | (exception bump: increment<br>mcs by 1)   |
| 0251 | 2072 | nft    | ldd | 72     | (new field test: load pc.                 |
| 0252 | 6561 |        | nzb | eidt   | if not zero go to eidt.                   |
| 0253 | 2204 |        | ldf | field1 | ow load 0310.                             |
| 0254 | 4171 |        | sti | 71     | store in output area.                     |
| 0255 | 5471 |        | aod | 71     | increment 1 by 1.                         |
| 0256 | 6565 |        | nzb | eidt   | go to eidt)                               |
| 0257 | 0310 | field1 |     | 310    | (code for field setting of 1)             |
| 0260 | 2071 | pd     | ldd | 71     | (punch data: load current                 |
| 0261 | 4204 |        | stf | 4      | value of 1. store forward<br>4 locations. |
| 0262 | 7500 |        | exf | 4104   | select punch.                             |
| 0263 | 4104 |        |     | fval   | punch data.                               |
| 0264 | 7303 |        | out | 0      | last word address + 1.                    |
| 0265 | 0000 |        |     |        | (determined during<br>execution)          |
| 0266 | 7073 |        | jpi | 73     | go to turt.                               |
| 0267 | 3000 | fval   |     | 3000   | first word address 1)                     |





# APPENDIX D

## cdc 160 memory map

| decimal | octal | contents  |
|---------|-------|---|
| 0000    | 0000  | program<br>can load at zero or<br>anywhere after 0077.<br>requires 267 octal<br>locations |
| 0511    | 0777  |   |
| 0512    | 1000  | mag tape input  |
| 1535    | 2777  |   |
| 1536    | 3000  | punch output  |
| 3584    | 7000  |   |
| 3585    | 7001  | leader-trailer  |
| 3684    | 7144  | code of 0200 octal  |
| 3685    | 7145  | available for use   |
| 4095    | 7777  |   |



## APPENDIX E

### SIMUL8S USER MANUAL CHANGES

It was originally intended to use this appendix to document the page changes made to references 1 and 2. As only five copies of each of the references were distributed, it was felt that the most expedient method of making the changes would be for the holders of the copies to seek out the updated master references and make the corresponding changes themselves. Accordingly, holders of references 1 and 2 are referred to the Director of the Computer Center who maintains custody of the updated references.



## LIST OF REFERENCES

1. Simulation 8 Series Assembler and Simulator Featuring PAL III Compatability, Decision Science, Inc., 1969
2. SIMUL8S USER GUIDE for IBM 360 Computer, Decision Science, Inc., 1969
3. PDP-8 Floating-Point System Programmer's Reference Manual, DEC-08-YQYB-D, 9th ed., Digital Equipment Corporation, 1969
4. Binary Loader. DEC-08-LBAB-D, Digital Equipment Corporation, 1971
5. Introduction to Programming, 2nd ed., v. 1, Digital Equipment Corporation, 1970
6. Control Data 160 Computer Programming Manual, Control Data Corporation
7. Martinsen, G. T., A Monitor for the PDP-8/S Computer, M. S. Thesis, United States Naval Postgraduate School, Monterey, 1970
8. Saber, G. W., A Fortran Compiler for the PDP-8 Computer, M. S. Thesis, United States Naval Postgraduate School, Monterey, 1971



# INITIAL DISTRIBUTION LIST

|  | No. Copies |
|--|------------|
| 1. Defense Documentation Center<br>Cameron Station<br>Alexandria, Virginia 22314   | 2          |
| 2. Library, Code 0212<br>Naval Postgraduate School<br>Monterey, California 93940   | 2          |
| 3. LTJG Alan B. Roberts, Code 53Ro<br>Department of Mathematics<br>Naval Postgraduate School<br>Monterey, California 93940   | 1          |
| 4. Mr. S. P. Tucker, Code 58Tc (PDP-8/S)<br>Department of Oceanography<br>Naval Postgraduate School<br>Monterey, California 93940  | 2          |
| 5. Assoc. Professor G. K. Poock, Code 55Pk (PDP-8/E)<br>Department of Operations Research/Administrative Sciences<br>Naval Postgraduate School<br>Monterey, California 93940 | 2          |
| 6. Mr. W. L. Landaker, Code 52Ec (CDC-160)<br>Department of Electrical Engineering<br>Naval Postgraduate School<br>Monterey, California 93940                                | 2          |
| 7. Professor D. G. Williams, Code 0211 (IBM System 360)<br>Director Computer Center<br>Naval Postgraduate School<br>Monterey, California 93940                               | 2          |
| 8. LT John W. Barnes, Jr., USN<br>U. S. Naval Ship Repair Facility<br>Box 34<br>FPO San Francisco, California 96651  | 1          |





## DOCUMENT CONTROL DATA - R &amp; D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

|   |  |   |                      |
|---|--|---|----------------------|
| 1. ORIGINATING ACTIVITY (Corporate author)<br>Naval Postgraduate School<br>Monterey, California 93940   |  | 2a. REPORT SECURITY CLASSIFICATION<br>Unclassified  |                      |
|   |  | 2b. GROUP   |                      |
| 3. REPORT TITLE<br>An INTERFACE for the PDP-8 Computer System Comprising Assembly, Compilation, Simulation, and PDP-8 Execution of Resulting Object Modules   |  |   |                      |
| 4. DESCRIPTIVE NOTES (Type of report and, inclusive dates)<br>Master's Thesis; June 1972  |  |   |                      |
| 5. AUTHOR(S) (First name, middle initial, last name)<br>John Winthrop Barnes, Jr.   |  |   |                      |
| 6. REPORT DATE<br>June 1972   |  | 7a. TOTAL NO. OF PAGES<br>43  | 7b. NO. OF REFS<br>8 |
| 8a. CONTRACT OR GRANT NO.   |  | 9a. ORIGINATOR'S REPORT NUMBER(S)   |                      |
| b. PROJECT NO.  |  |   |                      |
| c.  |  | 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)                 |                      |
| d.  |  |   |                      |
| 10. DISTRIBUTION STATEMENT<br>Approved for public release; distribution unlimited.  |  |   |                      |
| 11. SUPPLEMENTARY NOTES   |  | 12. SPONSORING MILITARY ACTIVITY<br>Naval Postgraduate School<br>Monterey, California 93940 |                      |
| 13. ABSTRACT<br><p>The design and implementation of the INTERFACE for the PDP-8 Computer System is described. The INTERFACE executes on an IBM System 360 and allows PAL III assembly, FORTRAN/8 compilation, and simulated execution of the resulting object code. If desired, object code is generated and processed for later execution on a PDP-8 computer.</p> |  |   |                      |



14

KEY WORDS

LINK A

LINK B

LINK C

ROLE

WT

ROLE

WT

ROLE

WT

PDP-8 Simulator  
PDP-8 Assembler  
PDP-8 Software Interface  
FORTRAN/8 Compiler









9  
m  
v  
it  
m



134939

Thesis  
B2297 Barnes  
c.1

An INTERFACE for the  
PDP-8 computer system  
comprising assembly,  
compilation, simulation,  
and PDP-8 execution of  
resulting object mod-  
ules.

Th

Thesis  
B2297 Barnes  
c.1

134939

An INTERFACE for the  
PDP-8 computer system  
comprising assembly,  
compilation, simulation,  
and PDP-8 execution of  
resulting objects mod-  
ules.

thesB2297

An INTERFACE for the PDP-8 computer syst



3 2768 002 01432 6

DUDLEY KNOX LIBRARY